WEEK - ~~10~~ 9

(Lab Pgm 10)

Inter Process Communication

Incorrect implementation of a producer and consumer.

```
class Q
{
        int n;
        synchronized int get()
        {
                System.out.println("Got : "+n);
                return n;
        }
        synchronized void put(int n)
        {
                this.n = n;
                System.out.println("Put : "+n);
        }
}
class Producer implements Runnable
 {
          Q q;
          Producer(Q q)
          {
                this.q = q;
                new Thread(this, "Producer").start();
          }
          public void run()
          {
                int i=0;
~~[illegible]~~       while (i < ~~[illegible]~~ 10)
                      {
                                q.put (i++);
                      }
             }
}
class Consumer implements Runnable
 {
            Q q;
            Consumer (Q q)
            {
                  this.q = q;
                  new Thread (this, "Consumer").start();
            }
            public void run()
            {
                   int i = 0;
                   while (i < 10)
                   {
                          int r = q.get();
                          i++;
                   }
            }
}
class PC
{
       public static void main (String args[])
       {
                Q q~~[illegible]~~ = new Q();
                new Producer(q);
                new Consumer(q);
                System.out.println("Press Control-C
                to stop.");
       }
```

}

O/P:

```
Press Control-C to stop.
Put: 0
Put: 1
Put: 2
Put: 3
Put: 4
Put: 5
Put: 6
Put: 7
Put: 8
Put: 9
Got: 9
Got: 9
Got: 9
Got: 9
Got: 9
Got: 9
Got: 9
Got: 9
Got: 9
Got: 9
```

Date __/__/__
Page ______

Correct implementation of a producer and consumer

```
class Q
{
    int n;
    boolean valueset = false;
    synchronized int get()
    {
        while(! valueSet)
        ~~try{~~ try
            {
                System.out.println ("\n
                Consumer waiting \n");
                wait();
            }
            catch(InterruptedException e)
            {
                System.out.println("
                InterruptedException caught ");
            }
            System.out.println("Got : "+n);
            valueSet= false;
            System.out.println (" \n Intimate
            Producer\n");
            notify();
            return n;
            }
            ~~synchronized void~~
    synchronized void put(int n)
    {
        while(valueSet)
```

gutdas

```
        try
        {
            System.out.println("\n
            Producer waiting \n");
            wait();
        }
        catch(InterruptedException e)
        {
            System.out.println("Interrupted
            Exception caught");
        }
        this.n = n;
        valueSet = true;
        System.out.println("Put : "+n);
        System.out.println("\nIntimate
        Consumer \n");
        notify();
    }
}
class Producer implements Runnable
{
    Q q;
    Producer(Q q)
    {
        this.q=q;
        new Thread(this, "Producer").start();
    }
    public void run()
    {
        int i = 0;
        while(i < 5)
        {
            q.put(i++);
```

```
                }
            }
}
class Consumer implements Runnable
{
        Q q;
        Consumer(Q q)
        {
            this.q = q;
            new Thread(this, "Consumer").start();
        }
        public void run()
        {
            int i=0; 5
            while(i<10)
            {
                int r = q.get();
                System.out.println(" consumed:
                 "+r);
                i++;
            }
        }
}
class PCFixed
{
    public static void main(String args[])
    {
            Q q = new Q();
            new Producer(q);
            new Consumer(q);
            System.out.println(" Press
            Control-C to stop.");
    }
}
```

O/P:

```
Press control-C to stop
Put : 0
Intimate Consumer
Producer waiting
Got : 0
Intimate Producer
Put : 1
Intimate Consumer
Producer waiting
consumed : 0
Got : 1
Intimate Producer
consumed : 1
Put : 2
Intimate Consumer
Producer waiting
Got : 2
Intimate Producer
consumed : 2
Put : 3
Intimate Consumer
Producer waiting
Got : 3
Intimate Producer
consumed : 3
Put : 4
Intimate Consumer
Got : 4
Intimate Producer
consumed : 4
```

Deadlock.

```
class A
{
    synchronized void foo(B b)
    {
        String name = Thread.currentThread
                        ().getName();
        System.out.println(name + " entered
          A.foo");
        try
        {
            Thread.sleep(1000);
        }
        catch(Exception e)
        {
            System.out.println("A Interrupted");
        }
        System.out.println(name + " trying
          to call B.last()");
        b.last();
    }
    void last()
    {
        System.out.println("Inside A.last");
    }
}
class B
{
    synchronized void bar(A a)
    {
        String name = Thread.currentThread
                        ().getName();
            System.out.println(name + "entered B.bar")
            try
            {
                Thread.sleep(1000);
            }
            catch(Exception e)
            {
                                                   B Interrupted
                System.out.println(" ~~Inside~~
                ~~A.last~~ ");
            }
            System.out.println( name + "trying
            to call A.last()");
            a.last();
        }
        void last()
        {
            System.out.println("Inside A.last");
        }
    }
    class Deadlock implements Runnable
    {
        A a= new A();
        B b= new B();
        Deadlock()
        {
            Thread.currentThread().setName("Main Th
            Thread t= new Thread(this,
              "RacingThread");
            t.start();
            a.foo(b);
            System.out.println("Back in main
        }
        public void run()
        {
            b.bar(a);
            System.out.println("Back
            in other thread");
        }
        public static void main (String
         args[])
        {
            new Deadlock();
        }
}
```

O/P:

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
Back in other thread
```

[illegible]
13/2/24